FIXED FREQUENCY CLOCK OUTPUT HAVING A VARIABLE HIGH FREQUENCY INPUT CLOCK AND AN UNRELATED FIXED FREQUENCY REFERENCE SIGNAL

BACKGROUND

[0001] In the prior art, a high frequency embedded phase lock loop (PLL) can be used to provide a fixed-frequency video clock input frequency. Additional crystals or oscillators may be required to provide the desired frequency or modify the output frequency. The input frequency is fixed. Each clock requires a separate PLL. The video frequency cannot be easily changed without impacting the entire system. PLLs consume considerable application specific integrated circuit (ASIC) real estate and crystals are expensive. Changing video frequencies may require board changes. For some applications, e.g. inline laser printing, the video frequencies must be calibrated to the print engine mechanism so board changes are not practical.

[0002] In the prior art, complicated tap-delay feedback loops are used. The delay elements require custom layout. The design requires real-time calibration to adjust for process, voltage, temperature (PVT) and PVT drift. Delay elements require complicated production testing procedures, and delay elements are not portable. A dithered input reference cannot be used and the output frequency spectrum cannot be easily smeared to reduce radio frequency interference (RFI). Due to the complex calibration and testing features, the design is large.

SUMMARY

[0003] The present invention generates very precise clock frequencies for applications that can tolerate a small degree of jitter but require exact long term frequencies, e.g. a video clock for a laser printer. Some subpixel jitter is acceptable, but the overall pixel rate must be exact and consistent. In some applications, the jitter may be desirable to smear the EMI spectrum. For example, if the high frequency input clock is modulated, the edges of the video clock will also be modulated yet remain within the jitter and frequency specification.

[0004] In its simplest form, the frequency synthesizer receives a dithered signal and a reference signal. From these two inputs, a constant frequency signal is generated. The dithered signal may be provided by an optional modulated analog PLL that receives a reference signal, Fref. The frequency synthesizer receives the output of the PLL Fdither, as an input frequency and the reference signal Fref to generate the video clock frequency signal Fout. Configuration registers transceive data and control with the PLL and the frequency synthesizer.

[0005] In one embodiment of the frequency synthesizer, a "simple predictor and corrector" receives the dithered signal Fdither and the reference signal Fref. From these inputs, it generates a "remove pulse" signal. An output generator, that receives the Fdither, Fref, and "remove pulse" signals, generates a "clear pulse" signal and the video frequency signal. Both the "simple predictor and corrector" and the output generator transceive data and control with the configuration registers.

[0006] In another embodiment of the frequency synthesizer, a predictor receives a dithered signal Fdither and a reference signal Fref. A corrector receives Fdither, Fref, and the output of the predictor. The output of the corrector indicates the fractional number of clocks to remove. An accumulator receives the corrector output and Fdither. From the accumulator output and Fdither, an output generator generates a constant frequency signal, Fout. All of the aforementioned blocks transceive data and control with the configuration registers.

BRIEF DESCRIPTION OF THE DRAWINGS

[0007] Figure 1 is a functional block diagram of a system 10 of the present invention.

[0008] Figure 2 is a functional block diagram of an embodiment of the frequency synthesizer 14 shown in Figure 1.

[0009] Figure 3 illustrates a flow process diagram 100 for the "simple predictor and corrector" 18 shown in Figure 2.

[0010] Figure 4 illustrates a flow process diagram 200 for the output generator 20 shown in Figure 2.

[0011] Figure 5 is an alternate functional block diagram of the frequency synthesizer 14.

[0012] Figure 6 is functional block diagram of another embodiment of a frequency synthesizer 40 according to the present invention.

[0013] Figure 7 is a flow process diagram 300 corresponding to the predictor 42 shown in Figure 6.

[0014] Figure 8 is a flow process diagram 400 corresponding to the corrector 44 shown in Figure 6.

DETAILED DESCRIPTION

[0015] The invention provides a method for generating very precise clock frequencies for applications that can tolerate a small degree of jitter but require exact long term frequencies, e.g. a video clock for a laser printer. Some subpixel jitter is acceptable, but the overall pixel rate must be exact and consistent. In some applications, the jitter may be desirable to smear the EMI spectrum. For example, if the high frequency input clock is modulated, the edges of the video clock will also be modulated yet remain within the jitter and frequency specification.

[0016] The invention takes advantage of a known fixed reference frequency and a high speed dithered clock. A known reference interval is used to calculate how many output clock edges (or pulses) should have occurred. By comparing the expected number of output transitions to the actual, it can correct the error suppressing or "swallowing" edges or pulses on the output. This statistically reduces the cumulative error to near zero.

[0017] In operation, the frequency synthesizer receives a dithered signal and a stable reference signal. From these two inputs, a constant frequency signal is generated.

[0018] Figure 1 is a functional block diagram of a system 10 of the present invention. An optional modulated analog PLL 12 receives a reference signal, Fref. A frequency synthesizer 14 receives the output of the PLL, Fdither, as an input frequency and the reference signal Fref to generate the video clock frequency signal Fout. Configuration registers 16 transceive data and control with the PLL 12 and the frequency synthesizer 14.

[0019] In an illustrative application, a laser printer controller requires an exact pixel rate of 20.12345 MHz. Subpixel resolution is required in the output signal. The video clock generated may be up to 12 times that frequency (12*20.12345 MHz = 241.4814 MHz). The rest of the controller system needs a ~250 MHz clock and a fixed I/O frequency of 48.000 MHz. The ~250 MHz clock will be further modulated to reduce EMI. The video frequency will be generated from clocks already in the system, e.g. 48.000 MHz and 500 +/- 10 MHz from the PLL. Fout cannot exceed half of the slowest frequency, e.g. 245 MHz. The video output should be the highest multiple of the video frequency possible while remaining lower than half of the slowest dithered input clock.

[0020] Figure 2 is a functional block diagram of an embodiment of the frequency synthesizer 14 shown in Figure 1. A "simple predictor and corrector" 18 receives the dithered signal Fdither and the reference signal Fref. From these inputs, it generates a "remove pulse" signal. An output generator 20, that receives the Fdither, Fref, and "remove pulse" signals, generates a "clear pulse" signal and the video frequency signal. Both the "simple predictor and corrector" 18 and the output generator 20 transceive data and control with the configuration registers 16.

[0021] Figure 3 illustrates a flow process diagram 100 for the "simple predictor and corrector" 18 shown in Figure 2. In step 110, the expected_value fractional counter is initialized. In step 120, it is determined whether a fixed reference edge has been

received. If no, step 120 is repeated. If yes, in step 130, the expected_value fractional counter is updated. In step 140, it is determined whether the actual value is greater than the integer of the expected value fractional counter. If no, step 120 is repeated. If yes, in step 150, the "remove pulse" flag is set.

[0022] Figure 4 illustrates a flow process diagram 200 for the output generator 20 shown in Figure 2. In step 210, the actual_value integer counter is initialized. In step 220, the output signal is initialized. In step 230, it is determined if there is a high frequency edge. If no, step 230 is repeated until a high frequency edge is detected. If yes, then in step 240, it is determined if remove_pulse flag has been set. If yes, then in step 250, the remove_pulse flag is cleared and the process returns to step 230. If no, then in step 260, the actual_value counter is incremented. In step 270, the output signal is toggled. and the process returns to step 230.

[0023] The embodiment shown in Figure 5 allows a single pulse to be removed each sample period (the lower fixed frequency clock defines the sample period). Figure 5 is an alternate functional block diagram of the frequency synthesizer 14. A first synchronizer 22 receives the I/O clock as reference input Fref and f_{dither} (Fin) as a clock input. An edge detector 24, connected to the output of the first synchronizer 22, receives as inputs a "twoedgedetect signal" and f_{dither}. A second synchronizer 26 receives an enable signal and f_{dither} as a clock input. A 24-bit adder 28 receives Referencecount[23:0] as an input. An Expected Count Latch 30 receives the output of the second synchronizer 26 as a clear input, f_{dither} as a clock input, the output of the 24-bit Adder 28 as data, and the output of the edge detector 24 as a load signal. The second input of the 24-bit Adder 28 and the output of the Expected Count Latch 30 are tied together. An Edge Counter 32 receives f_{dither} as a clock input and the output of the second synchronizer 26 as a clear signal. A comparator 34 receives the output of the Edge Counter 32 and the output of the Expected Count Latch 30 as inputs. The comparator 34 generates a rollover output, an A>B+1 signal, and an A>B signal.

[0024] ReferenceCount[23:0] represents the expected number of pulses to be counted each reference sample period. It represents a mixed number (integer and fraction).

[0025] When the reference frequency is very slow, we may wish to double it. Twoedgedetect indicates that sampling on both Fref clocks edges is required and not just a single edge.

[0026] When one needs to remove two pulses in-between reference edges, you may not want adjacent pulses as this increases jitter. To separate the second pulse removal from the first, the request is delayed a certain number of clocks.

[0027] SecondEdgeRemovalOffset accomplishes this by specifying the number of clocks to delay the second pulse removal. This is a configuration setting that comes from the microprocessor.

[0028] In operation, since the pulses are being "swallowed", the modulated (dithered) PLL must never go slower than the desired output frequency. In the present embodiment, the input frequency varies from 490 to 510 MHz. When we divide by 2, the output would be between 245 and 255 MHz, if no edges were swallowed. Since the desired frequency is 241.4814MHz and the slowest the input clock runs if 245 MHz, this is acceptable. Pulses will always be "swallowed" to attain the desired frequency.

[0029] If the dither is +/- 20 MHz, then the input could dip to 480 MHz that would lead to a 240 MHz output if no clocks were swallowed. This result is less than our desired frequency of 241.4814 MHz. So, we have to re-adjust the FSYNTH to give us 11*20.1234 MHz = 221.3574 MHz, so we remain below the input frequency. The desired frequency is changed by changing the value of the ReferenceCount signal.

[0030] Figure 6 is a functional block diagram of another embodiment of a frequency synthesizer 40 according to the present invention. A predictor 42 receives a dithered signal Fdither and a reference signal Fref. A corrector 44 receives Fdither, Fref, and the output of the predictor 42. The output of the corrector 44 indicates the fractional number of clocks to remove. An accumulator 46 receives the corrector output and Fdither. From the accumulator output and Fdither, an output generator 48 generates a

constant frequency signal, Fout. All of the aforementioned blocks transceive data and control with the configuration registers 16.

[0031] Figure 7 is a flow process diagram 300 corresponding to the predictor 42 shown in Figure 6. In step 310, the average number of high frequency (HF) clocks or dithered clocks is measured for n samples. N may be programmed by the user via the configuration registers. In step 320, the desired number of HF clocks per sample register is determined via the configuration registers. In step 330, the scale factor register value is determined via the configuration registers. In step 340, the difference between the measured clock periods and desired clock periods is determined. This difference is indicative of the average number of HF clocks to remove per sample period. In step 350, the average number of HF clocks to be removed is multiplied by the scale factor register value. This value indicates the average fractional number of HF clocks to remove each HF clock period.

[0032] Figure 8 is a flow process diagram 400 corresponding to the corrector 44 shown in Figure 6. In step 410, the error is measured from the last sample. In step 420, the error is scaled to a fractional error in terms of clocks/clock. In step 430, the scaled error is added to the average fractional number of HF clocks to remove per HF clock. The output represents a fractional number of clocks to remove each clock cycle.

[0033] In operation, the accumulator receives the fractional number of clocks to remove each clock cycle from the corrector. The output generator will remove a clock each time the accumulator output has a value greater than 1, e.g. has overflowed.

[0034] This embodiment generates a slower predictive value that indicates how much of each clock should be removed (on average). This should get the frequency close, but there can still be errors, so the 'Corrector' makes up for each sample's error. Instead of saying how many pixels to remove in a sample period, the output of the corrector is how much of a clock (a fractional amount) should be removed each clock. This way a large number of pulses can be removed in a very uniform way by having a large percentage of the pixel removed each clock. The accumulator has a running

total of the fractional part and any time the sum is 1 or greater (it overflows – no need for keeping the integer part of the addition), the next clock is removed.

[0035] Since the predictor can get close, but is not able to track faster changes and may not estimate exactly correctly. The corrector looks at each sample period and generates an error term. This error term is then normalized to the number of clocks per period (for example: remove 1 clock during the sample period, which usually has 8 clocks, so remove 1/8 of a clock each clock during the next sample period). This error term is added to the Predictor's so that the accumulator can generate the best position and number of clocks to remove as possible.

[0036] The embodiment disclosed in Figure 6 can handle removing many pulses during each sample period in a very nice way. This implementation provides more flexibility to chose frequencies since it is able to remove more than one pulse per sample period.